COMMUNICATIONS OPC WITH ECOSIMPRO

Jesús María Zamarreño Cosme
Dpto. Ingeniería de Sistemas y Automática (Fac. Ciencias) Universidad de Valladolid, jm@autom.uva.es

Abstract

This paper shows how to perform the integration of OPC with an EcosimPro simulation. The purpose is to obtain an OPC server that provides standard access to the simulation variables in real-time from any OPC-compliant application.

Keywords: OPC, Simulation in real-time, Integration, Communication.

1 INTRODUCTION

Usually, when anybody builds a simulation of any particular process in EcosimPro (or any other language), different objectives may exist like: design, trials in critical situations, tuning of controllers, etc. In all these areas, it is possible to perform the tasks through experiments from the same simulation environment. Nevertheless, a more flexible and open solution would be to use specific external tools that communicate in certain way with the simulation. For being able to do this, it is necessary to provide the simulation with some communication protocol. If the simulation code does not support any communication protocol, it is necessary to access the generated code and modify it. In this sense, one of the main advantages of EcosimPro is the possibility of using the generated C++ classes for integration with the communication protocol selected.

There are several choices for communicating the simulator with other applications (like HMI, SCADA, controllers, custom applications, etc), and the most common ones are data files, sockets, DDE; but, in any case, the result is usually very specific and not general enough for other cases. Another possibility is to use OPC (OLE for Process Control). OPC is an industrial standard designed for applications that need to interchange and share data in a control environment.

In the following sections, we show the steps that must be performed to come up with this integration, and, thus, converting any simulation in an OPC server accessible by any OPC client. First of all, some concepts of OPC will be provided, next, we will revise the possibilities of the C++ classes generated by EcosimPro, and, finally, we will describe the integration of the simulation into the OPC server.

2 OPC

OPC means OLE for Process Control. It is based on OLE/COM/DCOM Microsoft technology and it is an industrial standard that provides a common interface for communication that allows individual software components to interact and share data. OPC communication is performed in a client/server way. The OPC server is a data source (like a hardware device at the plant floor, but in our case, it will be a simulation) and any OPC-compliant application (the client) will have access to the server to read/write any variable served by the latter (Figure 1). One of the advantages of providing OPC capability to a simulation is that any OPC product acting as a client can access the simulation variables. Variables can be browsed and selected easily using natural names or tags.

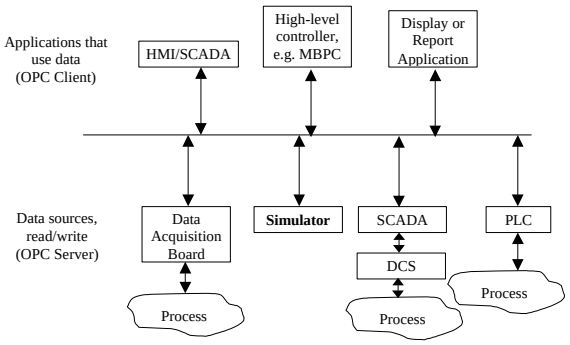

Figure 1: OPC communication between applications

Variables provided by an OPC server are structured in a hierarchical way. In first place, the OPC server can be located in a different node (i.e., computer) than the client. Thus, the root of an OPC server is specified as the node and the name of the server. Variables (or items) are included in a hierarchical namespace from the root. Clients classifies these item in groups, so the specification of any variable would be:

- ✓ Node
- ✓ Server_Name
- ✓ Path_through_namespace
- ✓ Item (it contains value, quality and timestamp)

The namespace is an important element in any OPC server since it allows the client to browse and find the desired variables. Figure 2 shows a particular example.

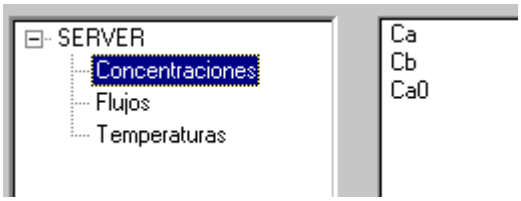

Figure 2: example of namespace

One limitation of OPC is that only runs on Microsoft platforms; specifically, Windows NT is recommended, though it can be installed on Windows 95 with DCOM extension, Windows 98, ME and now in recent Windows 2000.

Several tools exist for programming an OPC application; most of them consist in that they provide classes in C++ that facilitates the development of the application. This allows and facilitates a perfect integration with the C++ classes generated by EcosimPro.

3 C++ GENERATION BY ECOSIMPRO

C++ classes generated by EcosimPro allows its reusing for other tasks. The possibilities that provide are equal or even superior to that of the experiment concept in EcosimPro. Anything possible in an experiment can be done from a C++ program.

3.1 AUTOMATICALLY GENERATED CLASSES BY ECOSIMPRO

When an experiment is compiled, two C++ classes are automatically generated:

1. One partition class. It is coded in two files with name COMPONENT.PARTITION and extensions h and cpp.
2. One experiment class. It is coded in two files with name COMPONENT.PARTITION.EXPERIMENT and extensions h and cpp.

It will be necessary to use the INTEG_mt.lib library (provided by EcosimPro), that contains in particular the integration algorithms. It will also be necessary to make use of the definition files ecosim.h and MDL_common.h.

3.2 POSSIBILITIES FROM C++

3.2.1 Initialisation of the experiment

The first step is, from our C++ program, declaring an experiment object and making an initialisation:

```
model_simul_exp expe;
initEcosim(&expe);
```

From this point, the expe object can be utilized to perform any function allowed in an experiment.

3.2.2 Integration of the experiment (simulation)

In particular, it will be interesting to perform an integration of the model, so the system is simulated through time. This can be done in C++ through the INTEG_CINT() method of the experiment, that performs the integration during one communication interval defined in the CINT attribute of the experiment (we will assume that is given in hours). If we want that the simulation rate runs in real time, the code could be like the following (below). It is obvious that it is necessary that the integration time must be shorter than real time, so the rest of the time can be used for waiting (idle time). It is possible to make the simulation faster than real time introducing a new parameter acf (acceleration factor).

```
UINT ThreadExpe (LPVOID lparam)
{
 LARGE_INTEGER tbegin, tend, frec;
 double factor;
        // 'factor' is the ration between the
        // communication interval and calculation
        // time
 expe.INTEG_CINT();
        // first integration

 while(true)
        {
        QueryPerformanceCounter(&tbegin);
        if (expe.INTEG_CINT() == INTEG_END)
                break;
        QueryPerformanceCounter(&tend);

        QueryPerformanceFrequency(&frec);
        t_calculo = (double) (tend.QuadPart -
                tbegin.QuadPart) / (double)
                frec.QuadPart;

        factor = (expe.CINT)*3600.0/t_calculo;
        lmaxac = (lmaxac < factor) ? lmaxac : factor;

        if (factor>1)
                Sleep((unsigned long)((factor -
                        1)*t_calculo*1000/acf));
        }
        return 1;
}
```

3.2.3 Accessing the variables

Another interesting point is to access the simulation variables for being able to read values and

transferring them to other modules, as well as being able to write new values as desired. For performing the reading, the getValueReal method can be used. It must carry an argument: the name of the variable in EcosimPro. For example, for reading the Ca variable from the experiment, the code would be:

```
value = expe.getValueReal("Ca");
```

For writing a value in any variable, it can be done through the setValueReal method. The arguments are the name of the variable in EcosimPro and the value that we want to store:

```
expe.setValueReal("Fl",value);
```

These are the main aspects of manipulation of the experiment from C++ and provide us with a direct and easy access to the EcosimPro model.

4 INTEGRATION OF OPC AND ECOSIMPRO

Once OPC principles and how to employ the EcosimPro model from C++ have been explained, let show how to integrate both elements. For this purpose, in first place, it is convenient to have a "frame" written in C++ with all the functionality of an OPC server, but without the variables of the simulation neither the simulation. C++ classes generated by EcosimPro must be included within this frame and you must make proper calls and add some code at proper points in the program. For example, when the server starts, at the same time, you should declare the experiment object and make the initialisation as shown before. Other task would be to build the namespace depending on the variables of the simulation. Finally, and also important, is to establish how reading/writing is performed whenever any OPC client requires it. This is done using the methods seen at 3.2.3 section in the routine that manages communication requests from the clients.

Figure 3 shows a scheme of the steps that must be followed to build this OPC server.

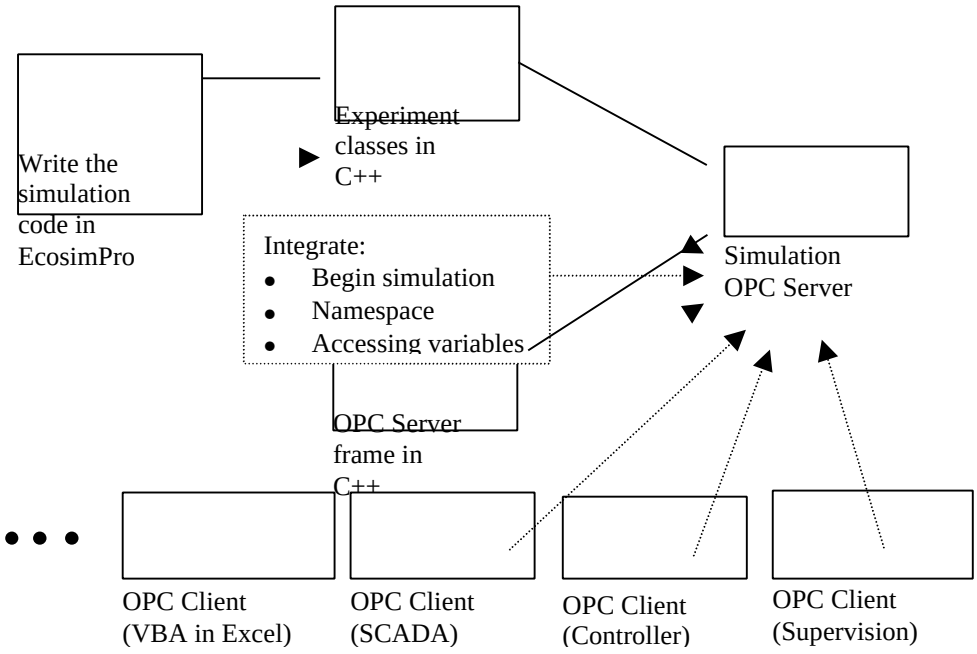

Figure 3: Procedure for generating an OPC server based on an EcosimPro simulation

5 INDUSTRIAL APPLICABILITY

OPC is today a real standard in data exchanging and sharing in the industrial field. Because of this, practically all of the present SCADAs are able to, acting like a client, acquire data through OPC. As explained in [1], one possible application is to perform the design and configuration of the SCADA without the need of the plant to exist, using the OPC server-simulator as the data source to make the design as well as checking the correct operation. Experiments in simulation with the SCADA may suggest a change in the parameters of the plant at an early stage. Once everything is working in simulation as desired, and the plant is physically built, only a reassignment of tags is necessary so they come, for example, from a data acquisition board (of course, with an OPC server).

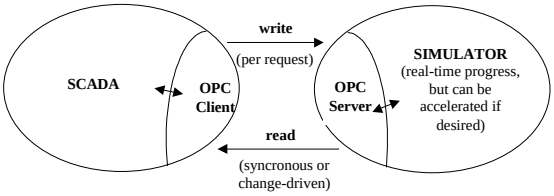

Figure 4: OPC link between an SCADA and the simulation

6 CONCLUSIONS

This paper has shown how to convert any EcosimPro model into an OPC server that contains the simulation running in real time. In this way, the simulation become open to communication requests from any program in a standard and flexible way. If the design is based on a standard OPC server frame, including the EcosimPro model into the project is a matter of hours following the procedure outlined in this paper.

Acknowledgement

The author express his gratitude for the support of the CICYT through the FEDER project "Supervisión y Control Optimizado de Procesos (TAP 1FD97-1690)".

References

[1] Acebes, L.F., Zamarreño J.M. (2001) "Diseño de un SCADA para una planta piloto usando un simulador OPC", *Automática e Instrumentación*, 316, pp. 76-80.

[2] Opc Task Force, (1998) "OPC Overview", *OPC Foundation*.

[3] Softing, (1999) "OPC Server Toolkit Programming Guide".

[4] Zamarreño, J.M., Acebes, L.F., Alvés, R. (2000) "OPC-based real time simulator: architecture and practical example", *41st SIMS Simulation Conference*.